

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images,
please do not report the images to the
Image Problems Mailbox.

BUNDESREPUBLIK DEUTSCHLAND



JC971 U S PTO
09/938752
06/24/01

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 100 65 416.9

Anmeldetag: 27. Dezember 2000

Anmelder/Inhaber: Siemens AG, München/DE

Bezeichnung: Verfahren zum Betrieb einer industriellen Steuerung

IPC: G 05 B 19/04

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 10. Juli 2001
Deutsches Patent- und Markenamt
Der Präsident
im Auftrag

Sieck

Beschreibung**Verfahren zum Betrieb einer industriellen Steuerung**

5 Die Erfindung bezieht sich auf ein Verfahren zum Betrieb einer industriellen Steuerung, insbesondere für Produktionsmaschinen.

10 Ferner bezieht sich die Erfindung auf eine industrielle Steuerung zur Durchführung des erfindungsgemäßen Verfahrens.

Es ist heutzutage üblich, sowohl für die speicherprogrammierbare Steuerung (SPS) als auch für die Bewegungssteuerung (MC) jeweils unterschiedliche hierarchische Ablaufebenen zu modellieren, denen Software-Tasks zur Steuerung des jeweiligen technischen Prozesses zugeordnet werden. Diese Tasks können Systemaufgaben erfüllen, sie können aber auch anwenderprogrammiert sein.

20 Aus DE 197 40 550 A1 ist es bekannt, dass Prozesssteuerungsfunktionalitäten der speicherprogrammierbaren Steuerungen "SPS" und Bewegungsfunktionalitäten von MC-Steuerungen in einem einheitlichen konfigurierbaren Steuerungssystem integriert werden können.

25 Diese SPS/MC-Integration geschieht in Form des Zusammenschaltens von SPS- und MC-Steuerungsbaugruppen. Bei einer solchen Ausführung der Integration wird aber keine optimale und effiziente Taskstruktur für die Gesamtheit der Steuerungsaufgaben erreicht. Außerdem werden bei dieser Art der Integration hauptsächlich die klassischen MC-Funktionalitäten, wie sie insbesondere für Werkzeugmaschinen relevant sind, unterstützt. Anforderungen an die Steuerung, wie sie aus dem Betrieb von Produktionsmaschinen bekannt sind, werden durch 30 diese Art des Zusammenschaltens von SPS- und MC-Steuerungsbaugruppen nicht optimal unterstützt.

Aus EP 0 735 445 A2 ist es bekannt, zum Betrieb einer Werkzeugmaschine oder eines Roboters einen gesonderten Wartebefehl (WAIT) zu gebrauchen. Der hier beschriebene Wartebefehl (WAIT) unterstützt aber insbesondere die Steuerung von Produktionsmaschinen noch nicht optimal.

- In der Anmeldung DE 19 93 19 33.2 wird vorgeschlagen, den Takt des Kommunikationssystems zwischen dem PC-System und den peripheren Geräten für einen Wechsel zwischen einem Echtzeitbetriebsprogramm und einem Nicht-Echtzeitbetriebsprogramm heranzunehmen. Hier ist es aber die Aufgabe dieser Taktabgreifung aus dem Kommunikationssystem, in einem industriellen Prozess einen möglichst reibungslosen Wechsel zwischen Echtzeit- und Nicht-Echtzeitanwendungen stattfinden zu lassen.
- Bei dieser Ausgestaltung wird der Grundtakt aber nur aus dem Takt des Kommunikationsmediums abgeleitet und er wird nur für den Wechsel des Betriebssystemmodus eines PC-Systems verwendet.
- Der Erfindung liegt daher die Aufgabe zugrunde, für jeweils unterschiedliche Steuerungsaufgaben und unterschiedliche Randbedingungen bzw. Anforderungen des zugrunde liegenden technischen Prozesses in einfacher Weise optimale Ausprägungen einer industriellen Steuerung zu erstellen, die sowohl SPS- als auch MC-Funktionalität zur Verfügung stellt und somit auch für die Steuerung von Produktionsmaschinen geeignet ist.
- Diese optimalen Ausprägungen werden prinzipiell zum einen durch ein einheitliches konfigurierbares Ablaufebenenmodell für die Steuerungs-Tasks der industriellen Steuerung erreicht und zum anderen durch Mechanismen (Wait_for_Condition-Befehl), die es einem Anwender ermöglichen, im Programmfluss auf beliebige Bedingungen zu warten und höherprior zu reagieren.

Von diesem Ansatz ausgehend wird die oben genannte Aufgabe dadurch gelöst, dass Mechanismen bereitgestellt sind, die es einem Anwender ermöglichen im Programmfluß auf eine beliebige Bedingung zu warten, wobei bei Erfülltsein der Bedingung der

- 5 Programmfluß unmittelbar fortgesetzt wird, bei Nichterfülltsein der Bedingung der Programmfluß solange angehalten wird, bis das Erfülltsein der Bedingung festgestellt wird, wobei beim Warten auf das Erfülltsein der Bedingung die Priorität der Bedingungsüberprüfung im Vergleich zur aktuellen Taskpriorität erhöht wird. Durch diesen Mechanismus wird es ermöglicht, eine zusammengehörige und geschlossene Aufgabenstellung in einem Stück Code eines Anwenderprogrammes auszudrücken, ohne dass weitere Mechanismen wie z.B. Event-Handler erforderlich sind. Ein Anwender kann somit in einem sequentiellen Programmablauf auf einer relativ niedrigen Prioritätsebene einer "Motion Task" durch Programmkonstrukte in seinem Programmfluß (Anwenderprogramm) das Warten auf hochpriore Ereignisse formulieren, ohne in ein anderes Programm wechseln zu müssen. Dadurch wird einerseits in der Steuerung Verwaltungs-Overhead vermieden, was direkt die System-Performance erhöht und andererseits aus programmiertechnischer Sicht das Lokalitätsprinzip unterstützt, wodurch z.B. das Debugging erleichtert wird.
- 20 25 Der beschriebene Mechanismus und der dazugehörige Befehl werden im weiteren als "wait_for_condition" bezeichnet.

Eine erste vorteilhafte Ausgestaltung der vorliegenden Erfindung liegt darin, dass nach dem Erfülltsein der Bedingung die folgende Programmsequenz bis zu einem expliziten Ende hochprior bearbeitet wird, wobei nach dem expliziten Ende der Programmsequenz die alte Taskpriorität wieder aufgenommen wird. Dadurch wird eine hohe Deterministik in der Sequenz "Warten auf externes Ereignis" und der "Aktion, die nach diesem Ereignis folgt", z.B. Korrekturbewegungen bei einer Druckmarkensynchronisation, erreicht. Ein Anwender hat somit die Möglichkeit sich in seinen Programmen temporär auf eine

hohe Prioritätsebene zu legen und dabei deterministische Vorgänge leicht und elegant beschreiben zu können. Anwendungsbeispiele sind z.B. Druckmarkensynchronisation und schnelle Bewegungsstart (beispielsweise nach Flankenwechsel).

5

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass für die Formulierung der Bedingungen Prozeßsignale und/oder interne Signale der Steuerung und/oder Variablen aus Anwenderprogrammen verwendet werden. Dadurch ist es dem 10 Anwender möglich, bei der Beschreibung der Bedingungen in einer einheitlichen Weise neben Anwenderprogrammvariablen auch direkt Systemzustände und Prozeßsignale zu verwenden.

15 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass die Bedingungen logische und/oder arithmetische und/oder beliebige funktionelle Verknüpfungen enthalten. Damit ist es dem Anwender möglich, innerhalb einer Anweisung komplexe Synchronisationsbeziehungen zu spezifizieren.

20 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass ein Anwenderprogramm für den Betrieb der Steuerung mehr als einen solchen Mechanismus enthält. Dadurch werden bei der Programmierung der Anwendungen die Flexibilität und die Möglichkeiten des Anwenders insbesondere hinsichtlich 25 der Beschreibung von Synchronisationsaktivitäten erhöht.

30 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass es mehrere Anwenderprogramme beim Betrieb der Steuerung geben kann, die diese Mechanismen enthalten. Dadurch werden bei der Programmierung der Anwendungen die Flexibilität und die Möglichkeiten des Anwenders insbesondere hinsichtlich der Beschreibung von Synchronisationsaktivitäten erhöht.

35 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass der jeweilige Mechanismus einem Anwender in einem Anwenderprogramm als übliches programmiersprachliches Kon-

strukt zur Verfügung steht. Der "wait_for_condition-Befehl", der diesen Mechanismus auslöst, kann somit von einem Anwender in den Anwenderprogrammen z.B. wie eine while-Schleife verwendet werden, wodurch die Programmerstellung sehr erleichtert wird.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass das Runtime-System der Steuerung ein Ablaufebenenmodell enthält, das mehrere Ablaufebenen unterschiedlichen Typs mit unterschiedlicher Priorität aufweist, wobei folgende Ablaufebenen vorgesehen sind:

a) eine Ebenengruppe mit synchron getakteten Ebenen, bestehend aus mindestens einer Systemebene und mindestens einer Anwenderebene, wobei die Ebenen dieser Ebenengruppe untereinander eine Priorisierung aufweisen können,

b) einer Anwenderebene für Systemexceptions

c) einer zeitgesteuerten Anwenderebene

d) einer ereignisgesteuerten Anwenderebene

e) einer sequentiellen Anwenderebene

f) einer zyklischen Anwenderebene,

wobei Anwenderebenen der Ebenengruppe a) optional synchron zu einer der Systemebenen der Ebenengruppe a) laufen können. Ein wesentlicher Vorteil dieser Schichtung liegt darin, dass die

Kommunikation zwischen den Tasks der Prozesssteuerung (SPS)

und denen der Bewegungssteuerung (MC) minimiert wird. Ein

weiterer Vorteil liegt darin, dass die Programmierung der Steuerungsaufgaben für die Prozesssteuerung und für die Bewegungssteuerung in einer einheitlichen Programmiersprache mit

einer einheitlichen Ersteloberfläche erfolgen kann und dass der Anwender ein für seine jeweiligen Anforderungen zugeschnittenes Ablaufebenenmodell flexibel erstellen kann.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass der Grundtakt des Ablaufebenenmodells aus einem internen Timer oder aus einem internen Takt eines Kommunikationsmediums oder aus einem externen Gerät oder von einer Größe, die zum technologischen Prozeß gehört abgeleitet wird. Dadurch kann sehr flexibel und sehr einfach der Grundtakt für das Ablaufebenenmodell abgeleitet werden. Dadurch, dass der Grundtakt für das Ablaufebenenmodell auch von einer Größe, die zum technologischen Prozess gehört, ableitbar ist, kann auf eine sehr einfache Weise eine direkte Rückkopplung aus dem technologischen Prozess zur Steuerung erhalten werden.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass die zeitgesteuerte Anwenderebene, die ereignisgesteuerte Anwenderebene, die sequentielle Ablaufebene, die zyklische Hintergrundebene und die Anwenderebene für System-exceptions optional sind. Der Anwender kann sich dadurch sehr flexibel eine Steuerung erstellen, die für seine konkreten vorliegenden Anforderungen sehr effizient ist und die die gerade benötigten Ablaufebenen enthält und somit keinen unnötigen Overhead beinhaltet.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass die synchronen Ebenen zum Grundtakt übersetzt und/oder untersetzt und/oder im Verhältnis 1:1 getaktet sind. Dadurch können die Ebenen jeweils sehr leicht zu einem Vielfachen des Grundtaktes getaktet werden oder aber auch jeweils zu einem reziproken Vielfachen getaktet werden. Ausgehend von einer gemeinsamen Ausgangsgröße können somit sehr einfach Übersetzungen, aber auch Untersetzungen für die jeweiligen Ebenen erreicht werden.

35 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass weitere priorisierende Schichtungen innerhalb der Ablaufebenen vorgesehen sind. Die Softwarestruktur der indus-

triellen Steuerung lässt sich dadurch optimal an die unterschiedlichen Steuerungsaufgaben bzw. an die Anforderungen des zugrunde liegenden technischen Prozesses anpassen. Somit lassen sich z.B. unterschiedliche Fehlerursachen unterschiedlichen Ebenen mit z.B. aufsteigender Priorität zuordnen.

Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass optional Anwendertasks beim Systemhochlauf und/oder beim Systemunterfahren durchlaufbar sind. Dadurch können beim Systemhochlauf z.B. Initialisierungsfunktionen angestoßen werden oder beim Systemunterfahren wird sichergestellt, dass die im System vorhandenen Achsen eine definierte Position einnehmen.

- 15 Eine weitere vorteilhafte Ausgestaltung der Erfindung liegt darin, dass in die Anwenderebenen Anwenderprogramme ladbar sind, die je nach Typ der Anwenderebene zyklusorientiert oder sequentiell programmiert sind. Dadurch kann der Anwender in seinen Anwenderprogrammen die Funktionalität der Steuerung sehr flexibel an die zugrunde liegenden Anforderungen des technischen Prozesses anpassen und er kann auch die Anwenderprogramme in unterschiedliche Anwenderebenen laden, um dadurch eine für seine jeweiligen Applikationen effektive Ausprägung der Steuerung zu erreichen. Ein weiterer Vorteil
- 20 liegt darin, dass der Anwender in ein einheitliches Ablaufebenenmodell bzw. Runtime-System einer industriellen Steuerung sowohl zyklusorientierte Anwenderprogramme als auch ereignisorientierte Anwenderprogramme laden kann. Ein Anwender kann somit nach unterschiedlichen Paradigmen programmierte
- 25 Programme (zyklusorientiert für SPS-Funktionalität und ereignisorientiert bzw. sequentiell für Bewegungsfunktionalität) sehr flexibel und konform in die Anwenderebenen des Ablaufebenenmodells hinzuladen.
- 30 35 Ein Ausführungsbeispiel der Erfindung ist in der Zeichnung dargestellt und wird im folgenden erläutert.

Dabei zeigen:

- 5 FIG 1 die wesentlichen Ablaufebenen einer klassischen
 speicherprogrammierbaren Steuerung,
- 10 FIG 2 die wesentlichen Ablaufebenen einer Bewegungssteue-
 rung,
- 15 FIG 3 schematische Darstellung einer industriellen Steue-
 rung,
- 20 FIG 4 das Ablaufebenenmodell der erfindungsgemäßen indus-
 triellen Steuerung,
- 25 FIG 5 ein Ausführungsbeispiel für das Zuladen von Anwen-
 derprogrammen in die Anwenderebenen,
- 30 FIG 6 ein Ausführungsbeispiel für den Gebrauch und den
 Mechanismus des Wait_for_Condition-Befehls, im Ab-
 laufebenenmodell der erfindungsgemäßen industriel-
 len Steuerung,
- 35 FIG 7 ein weiteres Ausführungsbeispiel für den Gebrauch
 und den Mechanismus des Wait_for_Condition-Befehls,
 im Ablaufebenenmodell der erfindungsgemäßen indus-
 triellen Steuerung,
- FIG 8 die syntaktische Beschreibung des
 Wait_for_Condition-Befehls in einem Syntaxdiagramm,
- FIG 9 ein Beispiel für die Formulierung einer Expression
 in programmiersprachlicher Notation und
- FIG 10 in einer Schemadarstellung Möglichkeiten, wie der
 Grundtakt für die industrielle Steuerung gewonnen
 wird.

In der Darstellung gemäß FIG 1 sind die wesentlichen Ablaufebenen einer klassischen speicherprogrammierbaren Steuerung (SPS), angeordnet nach ihrer Priorität, gezeigt. Der Prioritätsanstieg ist dabei durch einen Pfeil symbolisiert. In der 5 niederpriorsten Ebene werden, wie durch eine gestrichelte Linie angedeutet, zwei unterschiedliche Aufgaben, nämlich ein freier Zyklus, d.h. "Anwenderebene freier Zyklus" und eine Hintergrund-Systemebene, d.h. "Systemebene Hintergrund" abgewickelt. Der Hintergrund-Systemebene sind z.B. Kommunikationsaufgaben zugeordnet. Bei einer folgenden Anwenderebene, 10 bezeichnet als "Anwenderebene zeitgesteuert", ist der Aufruf- takt der Tasks bzw. der Programme dieser Ebene parametrierbar. Es erfolgt eine Überwachung dahingehend, ob die Bearbei- 15 tung eines Anwenderprogrammes dieser getakteten Ebene recht- zeitig abgeschlossen ist, bevor das Startereignis erneut auf- tritt. Läuft die Taktzeit ab, ohne dass das Anwenderprogramm der zugeordneten Ebene fertig abgearbeitet ist, wird eine entsprechende Task einer prioritätsmäßig übernächsten "Anwen- 20 derebene für asynchrone Fehler" gestartet. In dieser "Anwenderebene für asynchrone Fehler" kann der Anwender die Behandlung von Fehlerzuständen ausprogrammieren.

Auf die "Anwenderebene zeitgesteuert" folgt eine "Anwenderebene Events". Die Reaktion auf externe oder interne Ereignisse (Events) erfolgt innerhalb der "Anwenderebene Events". 25 Ein typisches Beispiel für ein solches Ereignis ist das Schalten eines binären bzw. digitalen Eingangs, wodurch typischerweise ein Ereignis ausgelöst wird. In einer "Systemebene hochprior" liegen die Aufgaben des Betriebssystems, welche 30 die Arbeitsweise der programmierbaren Steuerung (SPS) sicherstellen.

Die Darstellung gemäß FIG 2 zeigt die wesentlichen Ablaufebenen einer Bewegungssteuerung (MC). Auch hierbei sind die einzelnen Ebenen nach ihrer Priorität hierarchisch, wie durch 35 einen Pfeil symbolisiert, angeordnet. Eine "Systemebene Hintergrund" und eine "Anwenderebene sequentiell" haben eine gleiche Priorität, nämlich die niedrigste. Diese aufgabenmä-

Die Zugehörigkeit ist wie bei FIG 1 durch eine gestrichelte Linie symbolisiert. Die Tasks der "Anwenderebene sequentiell" werden zusammen mit den Tasks der "Systemebene Hintergrund" im Round-Robin-Verfahren abgearbeitet. Typische Tasks der 5 "Systemebene Hintergrund" sind z.B. solche für Kommunikationsaufgaben. In der "Anwenderebene sequentiell" laufen die vom Anwender programmierten Programmteile für die eigentliche Steuerungsaufgabe. Stößt die Steuerung in einem dieser Programmteile auf einen Bewegungs- oder Positionierbefehl, wird 10 ein "Suspend" gesetzt, d.h. das Anwenderprogramm wird an dieser Stelle unterbrochen. In diesem Fall wird ein Befehl synchron genutzt. Die Abarbeitung dieses Bewegungs- oder Positionierbefehls geschieht in einer höchstprioren "Systemebene getaktet". Ein jeder Lageregler oder Interpolator, der in der 15 "Systemebene getaktet" abläuft, führt diesen Bewegungs- bzw. Positionierbefehl aus. Nach Ausführung des Befehls wird in die "Anwenderebene sequentiell" zurückgesprungen und das durch "Suspend" unterbrochene Anwenderprogramm wird durch ein "Resume" an der gleichen Stelle fortgesetzt. Die "Systemebene 20 getaktet" enthält neben den schon erwähnten Lageregbern auch den Interpolationsteil der Steuerung.

Auf die niederpriorste Ebene setzt die "Anwenderebene Events" 25 auf. Hier sind solche Tasks untergebracht, die auf externe oder interne Ereignisse reagieren. Solche Ereignisse können beispielsweise Alarme sein.

In einer folgenden "Anwenderebene synchrogetaktet" laufen 30 synchron getaktete Anwender-Tasks ab, z.B. Reglerfunktionalitäten. Diese Tasks sind synchronisiert zu getakteten Systemfunktionen wie zum Beispiel Interpolator, Lageregler oder zyklische Buskommunikation.

In der Darstellung gemäß FIG 3 wird in Form eines Strukturbildes gezeigt, dass die Steuerung eines technischen Prozesses P1 über das Runtime-System RTS einer industriellen Steuerung erfolgt. Die Verbindung zwischen dem Runtime-System RTS 35 der Steuerung und dem technischen Prozess P1 geschieht bidi-

rektional über die Ein-/Ausgänge EA. Die Programmierung der Steuerung und damit das Festlegen des Verhaltens des Runtime-Systems RTS geschieht im Engineering-System ES. Das Engineering-System ES enthält Werkzeuge für die Konfigurierung, Projektierung und Programmierung für Maschinen bzw. für die Steuerung technischer Prozesse. Die im Engineering-System erstellten Programme werden über den Informationspfad I in das Runtime-System RTS der Steuerung übertragen. Bezuglich seiner Hardware-Ausstattung besteht ein Engineering-System ES üblicherweise aus einem Computersystem mit Graphikbildschirm (z.B. Display), Eingabehilfsmitteln (z.B. Tastatur und Maus), Prozessor, Arbeits- und Sekundärspeicher, einer Einrichtung für die Aufnahme computerlesbarer Medien (z.B. Disketten, CDs) sowie Anschlusseinheiten für einen Datenaustausch mit anderen Systemen (z.B. weiteren Computersystemen, Steuerungen für technische Prozesse) oder Medien (z.B. Internet). Eine Steuerung besteht üblicherweise aus Eingabe- und Ausgabeeinheiten, sowie aus Prozessor und Programmspeicher. Es ist auch vorstellbar, dass die Steuerung eines technischen Prozesses P1 über mehrere Runtime-Systeme RTS von industriellen Steuerungen erfolgt.

Die Darstellung gemäß FIG 4 zeigt das Ablaufebenenmodell der industriellen Steuerung. Die Priorisierung der Ebenen wird durch einen Pfeil in Richtung zur höchsten Priorität angedeutet. Die niederpriorsten Ebenen sind die "zyklische Anwenderebene" und die "sequentielle Anwenderebene". Diese beiden Ebenen laufen mit der gleichen Priorität. Deshalb sind diese Ebenen in der Darstellung gemäß FIG 4 durch eine gestrichelte Linie getrennt. Die "zyklische Anwenderebene" beinhaltet die "Background Task", die zykluszeitüberwacht ist. In der "sequentiellen Anwenderebene" werden die "Motion Tasks" durchlaufen. "Motion Tasks" sind nicht zykluszeitüberwacht und dienen im Wesentlichen zur Beschreibung sequentieller Abläufe. "Motion Tasks" werden quasiparallel abgearbeitet. Generell enthalten alle Anwenderebenen eine oder mehrere Tasks. Die Tasks nehmen die Anwenderprogramme auf. Die Tasks der

"zyklische Anwenderebene" und der "sequentiellen Anwenderebene" werden in einem gemeinsamen Round-Robin-Zyklus abgearbeitet.

- 5 Die nächstfolgende Ebene ist die "zeitgesteuerte Anwenderebene". Die Tasks dieser Ebene werden zeitgesteuert aktiviert. Die Zeitsteuerung ist einer Granularität von Millisekunden einstellbar. Auf die "zeitgesteuerte Anwenderebene" folgt die "ereignisgesteuerte Anwenderebene". In dieser Ebene werden
10 nach Erkennen eines User Interrupts so genannte "User Interrupt Tasks" aktiviert. User Interrupt Ereignisse können als logische Verknüpfung von Prozessereignissen und/oder internen Zuständen formuliert werden.
- 15 Die nächsthöhere Ebene ist die "Anwenderebene für System Exceptions". In dieser "Anwenderebene für System Exceptions" werden System Interrupts überwacht, bei deren Eintreffen so genannte "Exceptions", d.h. Ausnahmefallbehandlungen, generiert werden. In der "Anwenderebene für System Exceptions" gibt es z.B. folgende Tasks, die bei Auftreten eines entsprechenden System Interrupts aktiviert werden:
- 20 a) "Time Fault Task", die beim Ansprechen von Zeitüberwachungen aktiviert wird,
25 b) "Peripheral Fault Task", die z.B. bei Prozess- und Diagnosealarmen aktiviert wird, aber auch bei Stationsausfall oder Stationswiederkehr,
30 c) "System Fault Task", die bei allgemeinen Systemfehlern aktiviert wird,
d) "Program Fault Task", die bei Programmierfehlern (z.B. Division durch Null) aktiviert wird,
35 e) "Time Fault Background Task", die beim Ansprechen der Zykluszeitüberwachung der Background Task aktiviert wird und

f) "Technological Fault Task", die bei Technologiefehlern aktiviert wird.

5 Als nächstes folgt die Ebenengruppe "synchron getaktete Ebenen". Diese Ebenengruppe besitzt die höchste Priorität im Ab-
laufebenenmodell. Die einzelnen Ebenen dieser Ebenengruppe können untereinander weitere Priorisierungen aufweisen. Die
Ebenengruppe "synchron getaktete Ebenen" besteht aus minde-
10 stens einer Systemebene und mindestens einer Anwenderebene.
Die Systemebenen beinhalten die Systemfunktionen wie z.B. La-
geregler oder Interpolator. In die Anwenderebenen dieser Ebe-
nengruppe können von einem Anwender flexibel Anwenderprogra-
me (AP1 - AP4; FIG 5) zugeladen werden.

15 Für die Taktung der "synchron getakteten Ebenen" gibt es eine Reihe unterschiedlicher Taktgenerierungsmöglichkeiten. Der Grundtakt kann z.B. aus einem internen Timer (T1; FIG 10) kommen oder aus einem internen Takt (T3; FIG 10) eines Kommunikationsmediums (z.B. Profibus) oder aber der Takt kann auch aus einem Prozessereignis des technologischen Prozesses abgeleitet werden. Ein solches Prozessereignis kann z.B. die Taktrate (TG; FIG 10) eines Vorgangs an einer Produktionsma-
schine oder Verpackungsmaschine sein. Anwenderebenen der Ebe-
nengruppe "synchron getaktete Ebenen" können dabei basierend
20 auf dem Grundtakt getaktet werden, sie können aber auch syn-
chron zu einer der Systemebenen der Ebenengruppe "synchron
getaktete Ebenen" laufen. Die Anwendertasks dieser zu einer
Systemebene synchronen Anwenderebene haben somit eine syn-
chrone, d.h. deterministische Beziehung zu einer vom Anwender
flexibel festlegbaren Systemebene. Das hat den Vorteil, dass
deterministische Reaktionen auf Systemtasks (Systemtasks lau-
fen in den Systemebenen), die der Anwender in seinen Anwen-
dertasks programmiert hat, die in den Anwenderebenen der Ebe-
nengruppe "synchron getaktete Ebenen" laufen, vom System ga-
rantiert werden. Das heißt z.B., dass das System garantiert,
30 dass diese "synchrone Anwenderebene" entsprechend beispiel-
35

haft vor dem Interpolator aktiviert wird oder aber auch vor einer beliebigen anderen Systemfunktion.

Die "zeitgesteuerte Anwenderebene", die "ereignisgesteuerte Anwenderebene", die "sequentielle Anwenderebene", die "zyklische Anwenderebene" sowie die "Anwenderebene für System Exceptions" sind optional.

Die Task der "zyklischen Anwenderebene" (Background Task) ist zykluszeitüberwacht. Die "Motion Tasks" dagegen sind nicht zykluszeitüberwacht und dienen im wesentlichen zur Beschreibung sequentieller Abläufe. Das heißt, das vorliegende Ablaufebenenmodell unterstützt einen Anwender sowohl bei der Programmierung von sequentiellen Abläufen als auch bei der Ereignisprogrammierung. Es können somit synchrone Ereignisse als auch asynchrone Ereignisse durch die Programmierung erfasst werden. In die Anwenderebenen sind die vom Anwender erstellten Anwenderprogramme (AP1 - AP4; FIG 5) zuladbar. Die Anwenderprogramme AP1 bis AP4 werden üblicherweise mit Hilfe einer Programmierumgebung des Engineering-Systems (ES; FIG 3) erstellt.

Die Darstellung gemäß FIG 5 zeigt ein Ausführungsbeispiel für das Zuladen von Anwenderprogramm in die Anwenderebenen. FIG 5 zeigt exemplarisch eine Ausprägung von Anwenderebenen des Ablaufebenenmodells. Durch die drei Punkte am unteren Rand der Zeichnung ist dargestellt, dass auch noch weitere Anwenderebenen, aber auch Systemebenen vorhanden sein können. Die Priorisierung der Ebenen wird wie im Vorangegangenen durch einen Pfeil in Richtung zur höchsten Priorität angedeutet. Den Anwenderebenen werden die Anwenderprogramme AP1 bis AP4, am rechten Bildrand durch kleine Rechtecke angedeutet, zugeordnet. Die Zuordnung wird dargestellt durch Zuordnungspfeile ZP1 bis ZP4. In den Anwenderebenen befinden sich Tasks, die die zugeladenen Anwenderprogramme AP1 bis AP4 aufnehmen. Diese Tasks werden dann nach einer gewissen Strategie (z.B. sequentiell) durchlaufen bzw. abgearbeitet. Sie können weiter-

hin die Eigenschaft besitzen, dass sie laufzeitüberwacht sind.

Die Darstellung gemäß FIG 6 zeigt ein Ausführungsbeispiel für den Gebrauch und den Mechanismus des Wait_for_Condition-Befehls, im Ablaufebenenmodell der erfindungsgemäßen industriellen Steuerung. Der Wait_for_Condition-Befehl (in FIG 6 dargestellt als wait_for_cond()) wird in dieser Darstellung exemplarisch in der "sequentiellen Anwenderebene" verwendet.

Der Wait_for_Condition-Befehl wird in den vom Anwender erstellten "Motion Tasks" MT1 und MT2 verwendet, die Bestandteil der "sequentiellen Anwenderebene" sind. Die "Motion Tasks" MT1 und MT2 hängen in einem Round-Robin-Zyklus, dargestellt durch den Pfeil von MT1 nach MT2 und durch den abgewinkelten Pfeil von MT2 zu MT1.

Die drei Punkte innerhalb dieses Pfeiles deuten an, dass noch weitere "Motion Tasks" im Round-Robin-Zyklus hängen können. Die "Motion Task" MT1 enthält den Wait_for_Condition-Befehl "wait_for_cond(cond_1)", die "Motion Task" MT2 den Wait_for_Condition-Befehl

"wait_for_cond(cond_2)". Jeweils drei Punkte, die innerhalb von MT1 bzw. MT2 verwendet werden, deuten an, dass neben den beiden Wait_for_Condition-Befehlen und den drei Positionierbefehlen pos1() bis pos3() noch weitere Befehle in den "Motion Tasks" enthalten sein können.

Insgesamt besteht das in FIG 6 exemplarisch dargestellte Ablaufebenenmodell eines Runtime-Systems für eine industrielle Steuerung aus folgenden Ebenen (aufgezählt von niedrigster bis höchster Priorität): "zyklische Anwenderebene", "sequentielle Anwenderebene" (die Tasks dieser beiden Ebenen haben die gleiche Priorität, dargestellt durch die gestrichelte Linie zwischen diesen Ebenen), "zeitgesteuerte Anwenderebene", "ereignisgesteuerte Anwenderebene", "Anwenderebene für Systemexceptions", "synchron getaktete Anwenderebene 2", "synchron getaktete Anwenderebene 1", "synchron getaktete Systemebene 2" und als höchspriore Ebene eine "synchron getaktete Systemebene 1".

Die Arbeitsweise des Wait_for_Condition-Befehls wird bei-
spielhaft an "wait_for_cond(cond_1)" aus der "Motion Task"
MT1 gezeigt. Ist die "Motion Task" MT1 im Round-Robin-Zyklus
5 an der Reihe, werden die Befehle der "Motion Task" MT1 solan-
ge bedient, bis die Zeitscheibe abgelaufen ist, oder eine Un-
terbrechung kommt. Trifft dies zu, wird die "Motion Task" MT2
als nächste Task im Zyklus bedient, usw. Wird in der "Motion
Task" MT1 der wait_for_cond(cond_1)-Befehl abgearbeitet, wird
10 die Bedingung cond_1 überprüft. Ist cond_1 = true, also er-
füllt, dann wird sofort der nächstfolgende Befehl pos2() aus-
geführt und eventuell weitere in MT1 vorhandene Befehle suk-
zessive abgearbeitet, bis die Kontrolle an die nächste Task
abgegeben wird.

15 Ist die Bedingung cond_1 = false, also nicht erfüllt, wird
sofort die "Motion Task" MT1 unterbrochen und im Round-Robin-
Zyklus wird MT2 bedient. Die Bedingung cond_1 wird aber in
die "synchron getaktete Systemebene 2" eingehängt (angedeutet
20 durch den durchgehenden abgewinkelten Pfeil vom
wait_for_cond(cond_1)-Befehl zu der "synchron getakteten Sys-
temebene 2") und im Takt dieser Systemebene auf ihr Erfüllt-
sein überprüft. Ist die Bedingung cond_1 erfüllt, wird im
Round-Robin-Zyklus die aktuelle Task verdrängt, d.h. ihr wird
25 die Zeitscheibe entzogen und die Motion Task MT1 wird sofort
unmittelbar nach dem wait_for_cond(cond_1) mit dem Positio-
nierungsbefehl pos2() fortgesetzt. Durch den gestrichelten Pfeil
wird der Rücksprung aus der "synchron getakteten Systemebene
2" zum Positionierungsbefehl pos2(), d.h. zur "sequentiellen An-
30 wenderebene" angedeutet.

Dadurch, dass bei Nichterfülltsein der Bedingung des
Wait_for_Condition-Befehls die Überprüfung der Bedingung in
einer hochprioren "synchron getakteten Systemebene" stattfin-
35 det, und bei Erfülltsein der Bedingung sofort die unterbro-
chene "Motion Task" fortgesetzt wird, ist es einem Anwender
möglich, bei der Programmierung von Bewegungssequenzen extrem

zeitkritische Applikationen mit einfachen Sprachmitteln zu spezifizieren. Die Performance und Deterministik wird noch dadurch erhöht, dass beim Überprüfen der Bedingungen in den jeweiligen hochprioren "synchron getakteten Systemebenen" nur 5 aktuell anstehende Bedingungen eingehängt und berücksichtigt werden.

Der hier beschriebene Mechanismus erfordert auch keinen expliziten Event-Handler. Der große Vorteil aus Anwendersicht 10 liegt somit darin, dass der Anwender jetzt in einem sequentiellen Ablaufprogramm auf einer relativ niedrigen Prioritätsebene einer "Motion Task" in seinem Programmfluss hochpriore Ereignisse mit Hilfe von Programmkonstrukten formulieren kann und nicht in ein anderes Programm wechseln 15 muss, das er dann über andere Mechanismen (z.B. per Hand oder interruptgesteuert) auf synchrone Anwendertask abbilden muss, sondern er hat die Möglichkeit, in einem geschlossenen Anwenderprogramm diesen Zyklus "Warten auf hochpriores Ereignis" und "hochpriore Reaktion" auf dieses Ereignis in einem Programm 20 geschlossen zu formulieren.

Die Bedingungen, die in einem Wait_for_Condition-Befehl abgefragt werden, können vom Anwender sehr flexibel und elegant formuliert werden. So kann der Anwender zur Formulierung dieser Bedingungen Programmvariablen aus einem Anwenderprogramm 25 verwenden oder interne Größen der Steuerung oder er kann auch Prozesssignale referenzieren. Diese Größen können dann logisch, arithmetisch oder mit beliebigen Funktionen inhaltlich verknüpft werden, um daraus eine Bedingung zu formulieren. 30 Neben dem hochprioren Abfragen, ob die Bedingung erfüllt ist, kann man sich auch vorstellen, dass bei Erfülltsein der Bedingung auch noch dazugehöriger Programmcode, d.h. eine dahinterliegende Reaktion, die anwenderprogrammierbar ist, hochprior ausgeführt wird. Und dass erst nach der Ausführung 35 dieses Programmcodes der Rücksprung in die niederpriore Ebene erfolgt.

Die Darstellung gemäß FIG 7 zeigt ein erweitertes Ausführungsbeispiel für den Gebrauch und den Mechanismus des Wait_for_Condition-Befehls, im Ablaufebenenmodell der erfundsgemäßen industriellen Steuerung. Der Wait_for_Condition-Befehl (in FIG 7 ebenfalls dargestellt als wait_for_cond()) wird in dieser Darstellung exemplarisch in der "sequentiellen Anwenderebene" verwendet. Der Wait_for_Condition-Befehl wird in den vom Anwender erstellten "Motion Tasks" MT3 und MT4 verwendet, die Bestandteil der "sequentiellen Anwenderebene" sind. Die "Motion Tasks" MT3 und MT4 hängen in einem Round-Robin-Zyklus, dargestellt durch den Pfeil von MT3 nach MT4 und durch den abgewinkelten Pfeil von MT4 zu MT3. Die drei Punkte innerhalb dieses Pfeiles deuten an, dass noch weitere "Motion Tasks" im Round-Robin-Zyklus hängen können. Die "Motion Task" MT3 enthält den Wait_for_Condition-Befehl "wait_for_cond(cond_3)", die "Motion Task" MT4 den Wait_for_Condition-Befehl "wait_for_cond(cond_4)". Jeweils drei Punkte, die innerhalb von MT3 bzw. MT4 verwendet werden, deuten an, dass neben den beiden Wait_for_Condition-Befehlen und den Positionierbefehlen pos4() bis pos8() noch weitere Befehle in den "Motion Tasks" enthalten sein können. Durch die programmiersprachlichen Konstrukte "wait_for_cond()" und "end_wait_for_cond" wird eine Programmsequenz in den "Motion Tasks" geklammert. In der "Motion Task" MT3 sind die Befehle pos5() und pos6() auf diese Weise geklammert. Auch in der "Motion Task" MT4 wird die Verwendung von "wait_for_cond()" und "end_wait_for_cond" angedeutet. Durch jeweils 3 Punkte ist in der "Motion Task" MT4 skizziert, dass vor, innerhalb und nach dem "wait_for_cond() - end_wait_for_cond"-Konstrukt weitere Anweisungen vorhanden sein können.

Das in FIG 7 exemplarisch dargestellte Ablaufebenenmodell eines Runtime-Systems für eine industrielle Steuerung besteht wie in FIG 6 aus folgenden Ebenen (aufgezählt von niedrigster bis höchster Priorität): "zyklische Hintergrundebene", "sequentielle Anwenderebene" (die Tasks dieser beiden Ebenen haben die gleiche Priorität, dargestellt durch die gestrichelte

Linie zwischen diesen Ebenen), "ereignisgesteuerte Anwenderebene", "zeitgesteuerte Anwenderebene", "Anwenderebene für Systemexceptions", "synchron getaktete Anwenderebene 2", "synchron getaktete Anwenderebene 1", "synchron getaktete Systemebene 2" und als höchspriore Ebene eine "synchron getaktete Systemebene 1".

In FIG 7 wird die Arbeitsweise des Wait_for_Condition-Befehls mit einer dazugehörigen Programmsequenz gezeigt beispielhaft an "wait_for_cond(cond_3)" aus der "Motion Task" MT3 gezeigt. Das Überprüfen der Bedingung cond_3 und das Abarbeiten der dazugehörigen Programmsequenz (geklemmt zwischen "wait_for_cond(cond_3)" und "end_wait_for_cond") erfolgen dabei auf einer höherprioren Ebene des Ablaufebenenmodells. Die zu "wait_for_cond(cond_3)" dazugehörigen Programmsequenz wird durch die Abfolge der Befehle pos5() und pos6() gebildet.

Ist die "Motion Task" MT3 im Round-Robin-Zyklus an der Reihe, werden die Befehle der "Motion Task" MT3 solange bedient, bis die Zeitscheibe abgelaufen ist, oder eine Unterbrechung kommt. Trifft dies zu, wird die "Motion Task" MT4 als nächste Task im Zyklus bedient, usw. Wird in der "Motion Task" MT3 der "wait_for_cond(cond_3)" -Befehl abgearbeitet, so wird die Bedingung cond_3 überprüft. Ist cond_3 = true, also erfüllt, dann wird der normale programmablauf fortgesetzt, d.h. als nächstes wird der Befehl pos5() ausgeführt und eventuell weitere in MT3 vorhandene Befehle sukzessive abgearbeitet, bis die Kontrolle an die nächste Motion Task abgegeben wird.

Ist die Bedingung cond_3 = false, also nicht erfüllt, wird sofort die "Motion Task" MT3 unterbrochen und im Round-Robin-Zyklus wird MT4 bedient. Die Bedingung cond_3 und die Befehle pos5() und pos6() (als dazugehörige Programmsequenz) werden in der Priorität der "synchron getakteten Systemebene 2" bearbeitet (angedeutet durch den durchgehenden abgewinkelten Pfeil, ausgehend von der Klammer, die die Zusammengehörigkeit von wait_for_cond(cond_3), end_wait_for_cond und dazugehöri-

ger Programmsequenz ausdrückt, hin zu der "synchron getakteten Systemebene 2"). Im Takt dieser Systemebene wird cond_3 auf ihr Erfülltsein überprüft. Ist die Bedingung cond_3 erfüllt, wird mit der Priorität der "synchron getakteten Systemebene 2" die dazugehörige Programmsequenz (hier: die Abfolge der Befehle pos5() und pos6()) abgearbeitet. Durch den gestrichelten Pfeil wird der Rücksprung aus der "synchron getakteten Systemebene 2" zum Positionierbefehl pos7(), d.h. zur "sequentiellen Anwenderebene" angedeutet.

10

Dadurch, dass bei Nichterfülltsein der Bedingung des Wait_for_Condition-Befehls die Überprüfung der Bedingung in einer hochprioren "synchron getakteten Systemebene" stattfindet, und bei Erfülltsein der Bedingung sofort eine dazugehörige, vom Anwender erstellbare Programmsequenz auf dieser hochprioren Systemebene ausgeführt wird, können sogar extrem zeitkritische Applikationen mit einfachen Sprachmitteln spezifiziert und durchgeführt werden.

15 20 Ein möglicher Anwendungsfall ist die Druckmarkensynchronisation. Dabei geht es darum, eine Druckmarke auf einem Material hochprior zu erkennen. Beim Erkennen dieser Druckmarke wird typischerweise ein Istwert erfasst ("Latchen" z.B. eines Lage- oder Geberistwertes). Ausgehend von diesem erfassten Istwert wird ein Korrekturwert berechnet, der dem System als überlagerte Bewegung aufgeprägt wird. Der Vorgang Istwert-Erkennung, Korrekturwert-Berechnung und Durchführung der überlagerten Bewegung muß in einer deterministischen Zeitdauer erfolgen. Deswegen muß dieser Vorgang hochprior stattfinden.

25

30 35 Ein weiterer Anwendungsfall ist der "schnelle Bewegungsstart". Hier geht es darum, z.B. Flankenwechsel sehr schnell zu erkennen und daran sofort anschließend einen Bewegungsstart (z.B. Positionierbewegung) zu beginnen. Die Deterministik Erkennen eines Ereignisses und Auslösen von Folgeaktionen ist entscheidend für die Produktivität einer Maschine. Bei Produktionsmaschinen müssen solche zyklischen Vorgänge in ei-

ner deterministischen Zeit, z.B. <100 ms oder <50 ms erfolgen. Beim Abarbeiten der Tasks auf einer normalen Hintergrundebene kann diese Deterministik nicht garantiert werden. Der beschriebene Mechanismus ist besonders für einen Einsatz bei Maschinen geeignet, die periodische Maschinenzyklen aufweisen.

Die Performance wird noch dadurch erhöht, dass beim Überprüfen der Bedingungen in den jeweiligen hochprioren "synchron getakteten Systemebenen" nur aktuell anstehende Bedingungen eingehängt und berücksichtigt werden.

Der hier beschriebene Mechanismus erfordert, wie schon in FIG 6 erwähnt, keinen expliziten Event-Handler. Der große Vorteil aus Anwendersicht liegt somit darin, dass der Anwender jetzt in einem sequentiellen Ablaufprogramm auf einer relativ niedrigen Prioritätsebene einer "Motion Task" in seinem Programmfluss hochpriore Ereignisse mit Hilfe von Programmkonstrukten formulieren kann und nicht in ein anderes Programm wechseln muss, das er dann über andere Mechanismen (z.B. per Hand oder interruptgesteuert) auf synchrone Anwendertask abbilden muss, sondern er hat die Möglichkeit, in einem geschlossenen Anwendungsprogramm diesen Zyklus "Warten auf hochpriores Ereignis" und "hochpriore Reaktion" auf dieses Ereignis in einem Programm geschlossen zu formulieren.

Der Wait_for_Condition-Befehl kann vom Anwender sehr flexibel und einfach eingesetzt werden, da er als normales programmiersprachliches Konstrukt zur Verfügung steht. Auch die Formulierung der Bedingungen ist für einen Anwender flexibel und einfach. So kann der Anwender zur Formulierung dieser Bedingungen Programmvariablen aus einem Anwendungsprogramm verwenden oder interne Größen der Steuerung oder er kann auch Prozesssignale referenzieren. Diese Größen können dann logisch, arithmetisch oder mit beliebigen Funktionen inhaltlich verknüpft werden, um daraus eine Bedingung zu formulieren.

Durch das Wait_for_Condition-Konstrukt hat ein Anwender die Möglichkeit in normalen Anwenderprogrammen für Bewegungssequenzen, ein Anwenderprogramm temporär auf eine höhere Prioritätsebene zu legen, um deterministische Vorgänge garantieren zu können.

Darstellung gemäß FIG 8 zeigt das programmiersprachliche Konstrukt des wait_for_condition-Mechanismus als Syntaxdiagramm. Die terminalen Elemente sind dabei mit abgerundeten Ecken dargestellt: "WAITFORCONDITION", "WITH", "DO", "END_WAITFORCONDITION" und ";". Als Rechtecke sind die nicht-terminalen Elemente dargestellt: "Expressions-Bezeichnung", "SCHALTER" und "ANWEISUNGSTEIL". Die Elemente "WITH" und "SCHALTER" sind optional.

Darstellung gemäß FIG 9 zeigt die Verwendung des Wait_for_Condition-Konstruktes in einem Programmablauf. Im oberen Teil von FIG 9 ist die Formulierung der Bedingung "my-Expression" dargestellt, im unteren Teil ist gargestellt, wie diese Bedingung in einem Wait_for_Condition-Konstrukt verwendet wird.

Darstellung gemäß FIG 10 zeigt in einer Schemadarstellung Möglichkeiten, wie der Grundtakt für die industrielle Steuerung gewonnen wird. FIG 10 zeigt exemplarisch eine Kommunikationstopologie, in die die Steuerung S integriert ist. Die Steuerung S ist durch ein Rechteck dargestellt. Durch eine Anschlussleitung A2 ist die Steuerung S mit dem Bus B1 verbunden, an dem über eine Anschlussleitung A1 das externe Gerät EG hängt. Über den Bus B2 erfolgt die Verbindung zum technischen Prozess P2. Der technische Prozess P2 ist am unteren Bildrand durch ein Rechteck dargestellt. Über die Anschlussleitung A3 ist die Steuerung S mit dem Bus B2 verbunden, der wiederum über die Anschlussleitung A4 die Verbindung zum technischen Prozess P2 herstellt.

Die Generierung für den Grundtakt der Steuerung S kann aus unterschiedlichen Taktquellen erfolgen. So z.B. aus einer internen Taktquelle, dargestellt durch den internen Timer T2 der Steuerung S oder auch durch eine externe Taktquelle wie 5 z.B. den Timer T1, der zum externen Gerät EG gehört. Als externe Taktquelle kann aber auch der Grundtakt eines Kommunikationsmediums dienen. Wenn der Bus B2 z.B. durch einen äquidistanten Profibus realisiert wird, dann kann der Takt für die Steuerung aus dem Grundtakt dieses Busses gewonnen werden. In FIG 10 ist dies dargestellt dadurch, dass der Timer 10 T3 direkt an der Anschlussleitung A3 positioniert ist, und diese Anschlussleitung A3 stellt die Verbindung zum Bus B2 her. Die Steuerung hängt somit als Slave am Bus und kann direkt den Bustakt verwenden. Weiterhin kann als externe Taktquelle ein Taktgeber TG dienen, der im technischen Prozess P2 integriert ist. Ein Taktgeber TG in einem technischen Prozess kann z.B. der Arbeitstakt einer Produktionsmaschine oder Verpackungsmaschine sein. In der Darstellung gemäß FIG 10 sind 15 als Kommunikationsmedien beispielhaft Busverbindungen dargestellt. Es können aber als Kommunikationsmedien aber auch Ring-, Stern- oder andere Verbindungsarten gewählt werden, auch wireless-Verbindungen. Aus diesen Verbindungssystemen kann dann der oben genannte Grundtakt abgeleitet werden.

Patentansprüche

1. Verfahren zum Betrieb einer industriellen, mit einem Runtime-System (RTS) ausgestatteten Steuerung (S), insbesondere
5 für Produktionsmaschinen,

dadurch gekennzeichnet,
dass Mechanismen bereitgestellt sind, die es einem Anwender ermöglichen im Programmfluß auf eine beliebige Bedingung zu warten, wobei bei Erfülltsein der Bedingung der Programmfluß
10 unmittelbar fortgesetzt wird, bei Nichterfülltsein der Bedingung der Programmfluß solange angehalten wird, bis das Erfülltsein der Bedingung festgestellt wird, wobei beim Warten auf das Erfülltsein der Bedingung die Priorität der Bedingungsüberprüfung im Vergleich zur aktuellen Taskpriorität erhöht wird.

2. Industrielle Steuerung nach Anspruch 1,

dadurch gekennzeichnet,
dass nach dem Erfülltsein der Bedingung die folgende Programmsequenz bis zu einem expliziten Ende hochprior bearbeitet wird, wobei nach dem expliziten Ende der Programmsequenz die alte Taskpriorität wieder aufgenommen wird.

3. Industrielle Steuerung nach Anspruch 1 oder 2,

25 dadurch gekennzeichnet,
dass für die Formulierung der Bedingungen Prozeßsignale und/oder interne Signale der Steuerung und/oder Variablen aus Anwenderprogrammen (AP1 - AP4) verwendet werden.

30 4. Industrielle Steuerung nach einem der vorstehenden Ansprüche,

dadurch gekennzeichnet,
dass die Bedingungen logische und/oder arithmetische und/oder beliebige funktionelle Verknüpfungen enthalten.

5. Industrielle Steuerung nach einem der vorstehenden Ansprüche,

d a d u r c h g e k e n n z e i c h n e t,

dass ein Anwenderprogramm (AP1 - AP4) für den Betrieb der

5 Steuerung mehr als einen solchen Mechanismus enthält.

6. Industrielle Steuerung nach einem der vorstehenden Ansprüche,

d a d u r c h g e k e n n z e i c h n e t,

10 dass es mehrere Anwenderprogramme (AP1 - AP4) beim Betrieb der Steuerung geben kann, die diese Mechanismen enthalten.

6 7. Industrielle Steuerung nach einem der vorstehenden Ansprüche,

15 d a d u r c h g e k e n n z e i c h n e t,

dass der jeweilige Mechanismus einem Anwender in einem Anwenderprogramm (AP1 - AP4) als übliches programmiersprachliches Konstrukt zur Verfügung steht.

20 8. Industrielle Steuerung zur Durchführung des Verfahrens nach einem der vorstehenden Ansprüche,

d a d u r c h g e k e n n z e i c h n e t,

dass das Runtime-System (RTS) der Steuerung (S) ein Ablaufebenenmodell enthält, das mehrere Ablaufebenen unterschiedlichen Typs mit unterschiedlicher Priorität aufweist, wobei folgende Ablaufebenen vorgesehen sind:

a) eine Ebenengruppe mit synchron getakteten Ebenen, bestehend aus mindestens einer Systemebene und mindestens einer Anwenderebene, wobei die Ebenen dieser Ebenengruppe untereinander eine Priorisierung aufweisen können,

30 b) einer Anwenderebene für Systemexceptions,

35 c) einer zeitgesteuerten Anwenderebene

d) einer ereignisgesteuerten Anwenderebene

e) einer sequentiellen Anwenderebene

f) einer zyklischen Anwenderebene,

5 wobei Anwenderebenen der Ebenengruppe a) optional synchron zu einer der Systemebenen der Ebenengruppe a) laufen können.

9. Industrielle Steuerung nach Anspruch 8,

d a d u r c h g e k e n n z e i c h n e t,

10 dass der Grundtakt des Ablaufebenenmodells aus einem internen Timer (T2) oder aus einem internen Takt (T3) eines Kommunikationsmediums oder aus einem externen Gerät (EG) oder von einer Größe (TG), die zum technologischen Prozeß (P1, P2) gehört abgeleitet wird.

15

10. Industrielle Steuerung nach Anspruch 8 oder Anspruch 9,

d a d u r c h g e k e n n z e i c h n e t,

dass die zeitgesteuerte Anwenderebene, die ereignisgesteuerte Anwenderebene, die sequentielle Ablaufebene, die zyklische Hintergrundebene und die Anwenderebene für Systemexceptions optional sind.

11. Industrielle Steuerung nach Anspruch 8, 9 oder 10,

d a d u r c h g e k e n n z e i c h n e t,

25 dass die synchronen Ebenen zum Grundtakt übersetzt und/oder untersetzt und/oder im Verhältnis 1:1 getaktet sind.

12. Industrielle Steuerung nach Anspruch 8, 9, 10 oder 11,

d a d u r c h g e k e n n z e i c h n e t,

30 dass weitere priorisierende Schichtungen innerhalb der Ablaufebenen vorgesehen sind.

13. Industrielle Steuerung nach Anspruch 8, 9, 10, 11 oder 12,

35 d a d u r c h g e k e n n z e i c h n e t,
dass optional Anwendertasks beim Systemhochlauf und/oder beim Systemunterfahren durchlaufbar sind.

14. Industrielle Steuerung nach Anspruch 8, 9, 10, 11, 12 o-
der 13,

d a d u r c h g e k e n n z e i c h n e t,

dass in die Anwenderebenen Anwenderprogramme (AP1 - AP4) lad-

5 bar sind, die je nach Typ der Anwenderebene zyklusorientiert
oder sequentiell programmiert sind.

Zusammenfassung**Verfahren zum Betrieb einer industriellen Steuerung**

- 5 Es werden Mechanismen zum Betrieb einer industriellen, mit einem Runtime-System (RTS) ausgestatteten Steuerung (S), insbesondere für Produktionsmaschinen bereitgestellt sind, die es einem Anwender ermöglichen im Programmfluss auf eine beliebige Bedingung zu warten, wobei bei Erfülltsein der Bedingung der Programmfluss unmittelbar fortgesetzt wird, bei
- 10 Nichterfülltsein der Bedingung der Programmfluss solange an gehalten wird, bis das Erfülltsein der Bedingung festgestellt wird, wobei beim Warten auf das Erfülltsein der Bedingung die Priorität der Bedingungsüberprüfung im Vergleich zur aktuellen Taskpriorität erhöht wird. Bei Erfülltsein der Bedingung wird eine definierte Programmsequenz bis zu einem expliziten Ende hochprior bearbeitet wird, wobei nach dem expliziten Ende der Programmsequenz die alte Taskpriorität wieder aufgenommen wird.
- 15

20

FIG 6

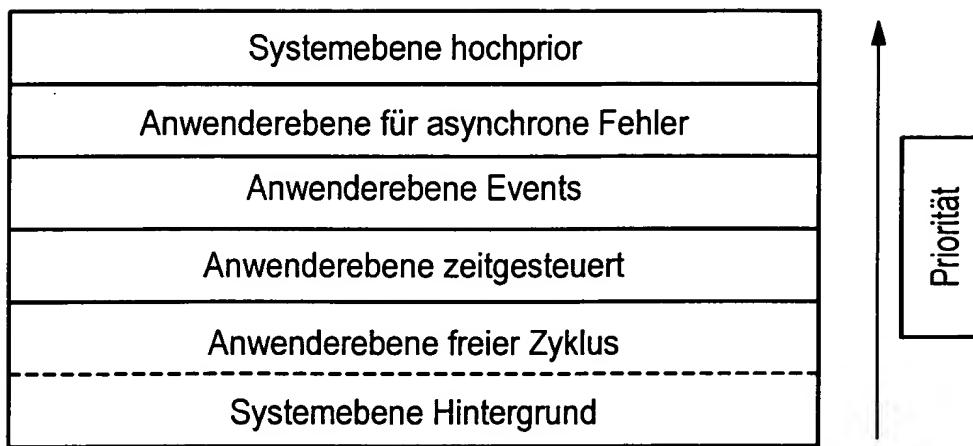


FIG 1

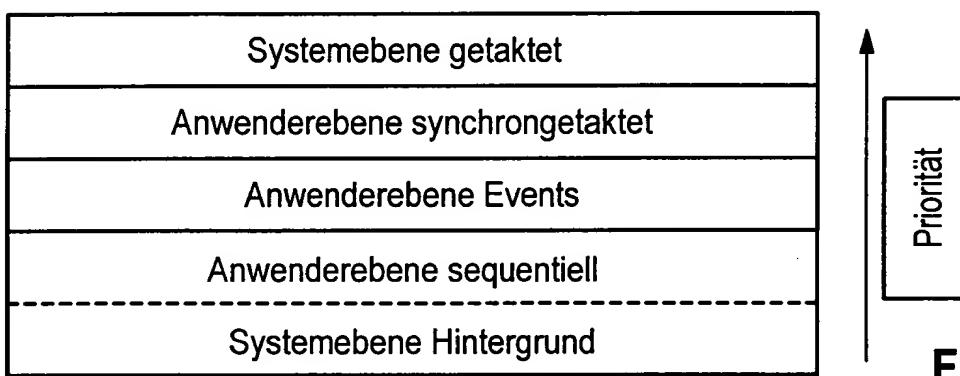


FIG 2

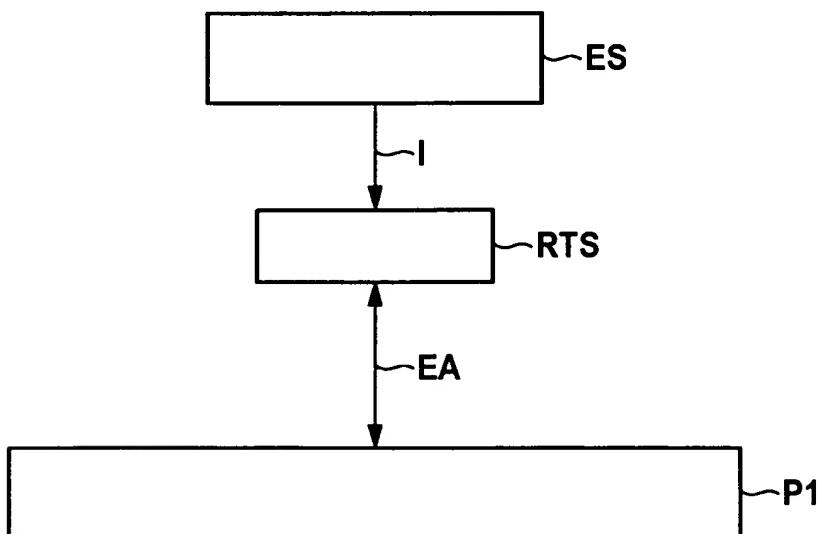


FIG 3

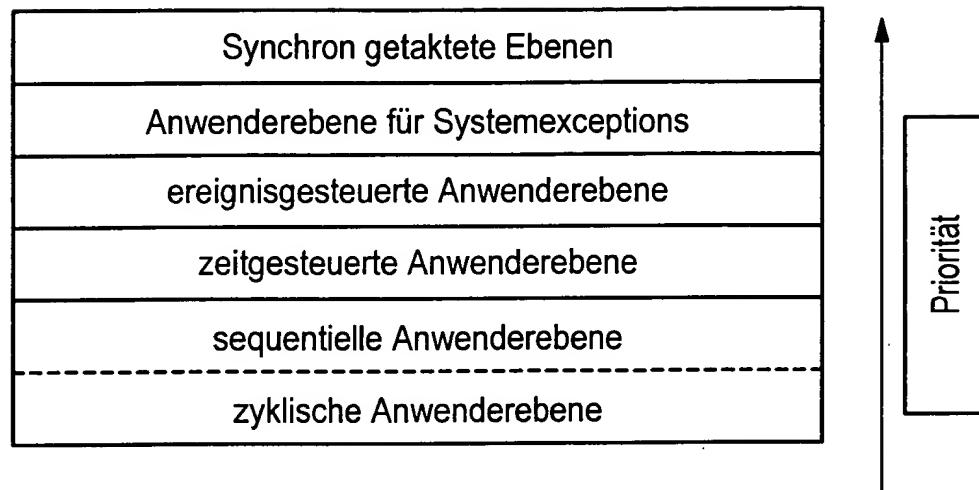


FIG 4

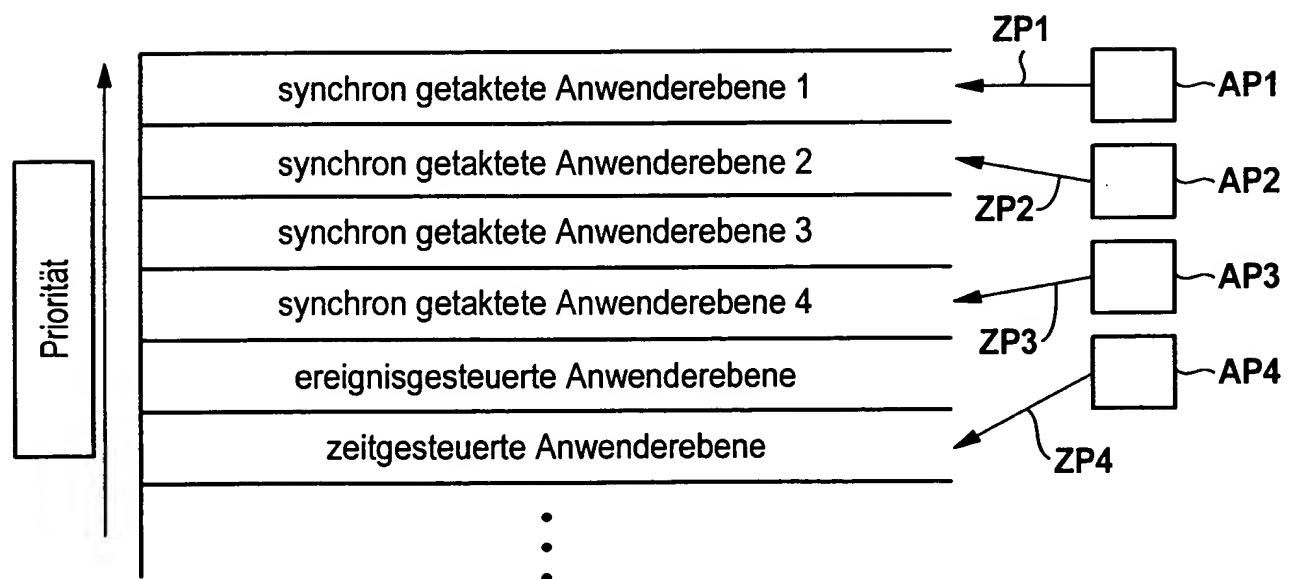


FIG 5

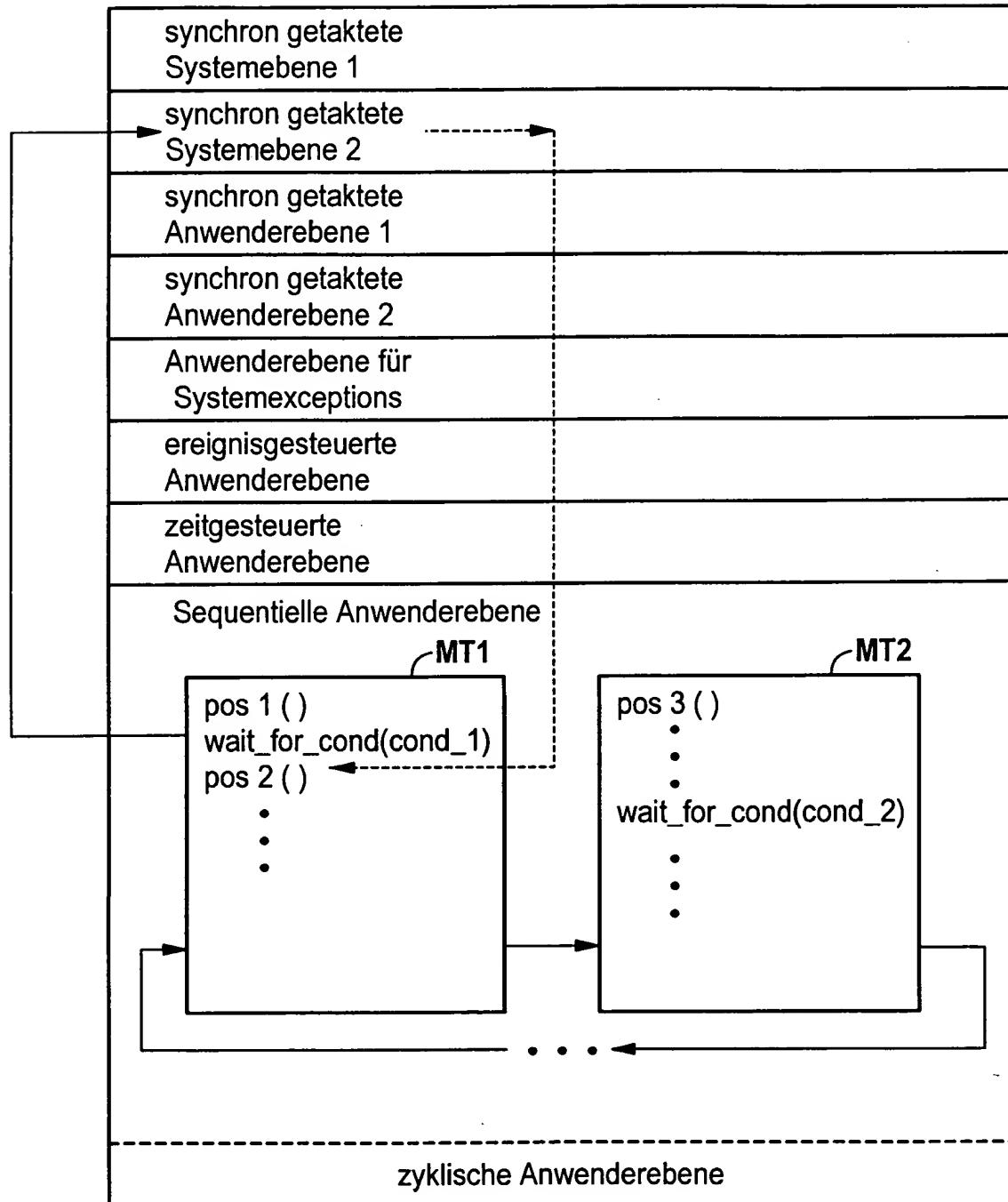


FIG 6

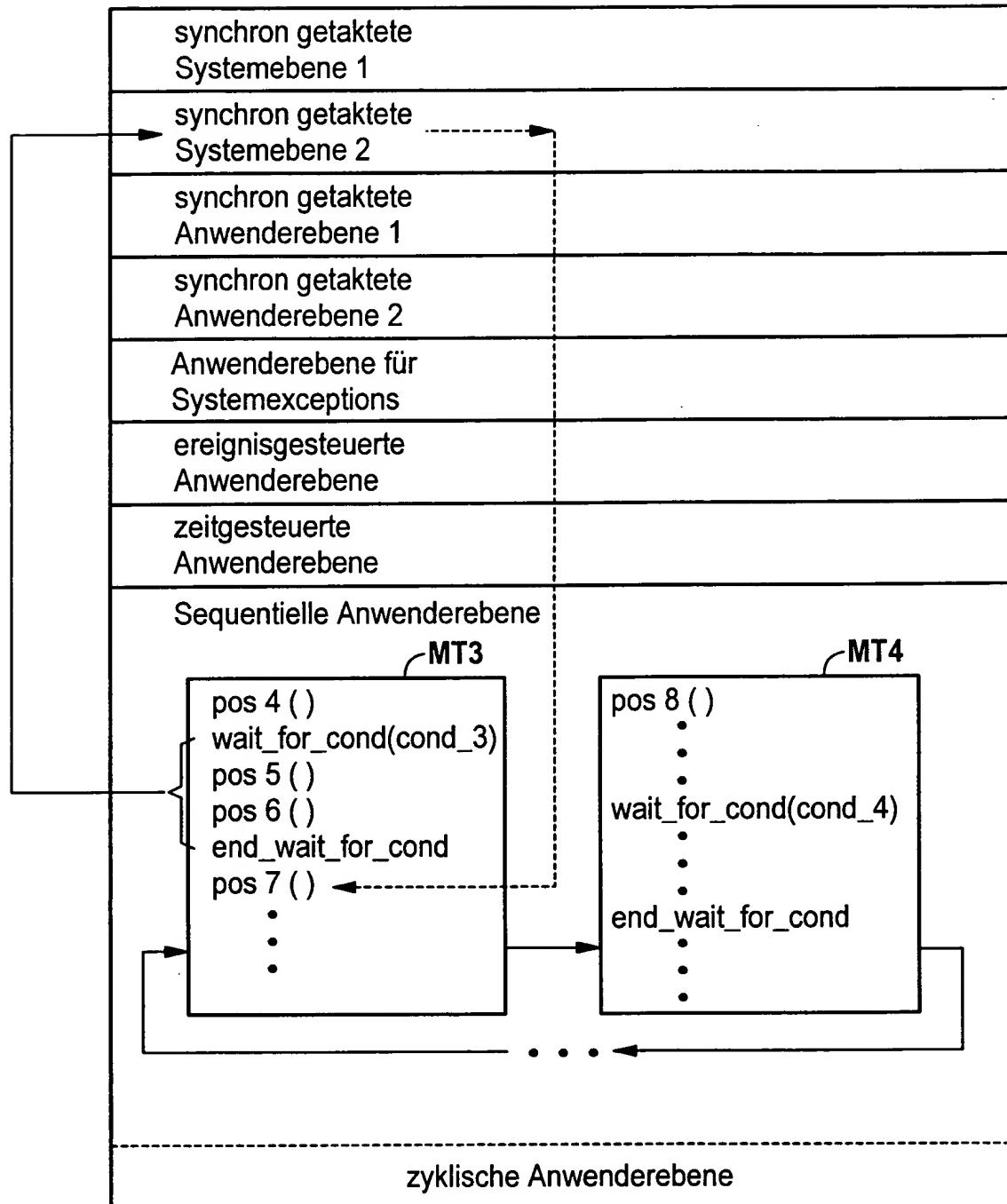


FIG 7

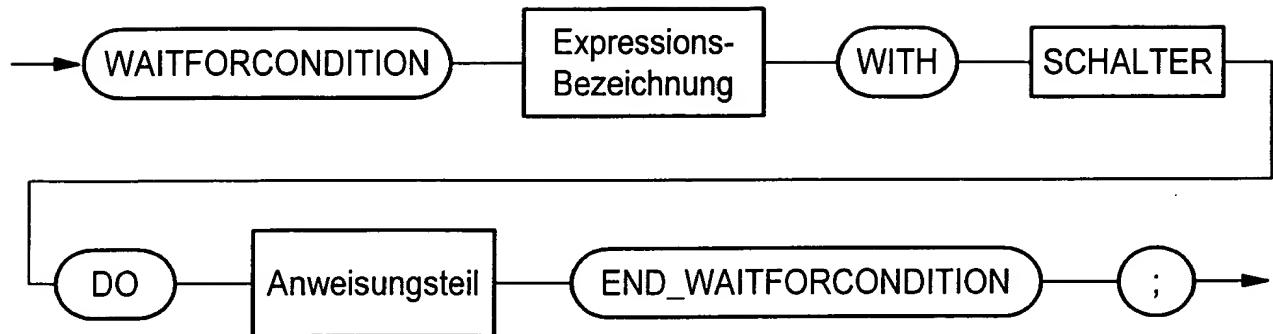


FIG 8

IMPLEMENTATION

```

EXPRESSION myExpression
//Vereinbarung lokaler Variablen möglich
VAR
    myLoc : WORD
END_VAR
    myLoc :=%IW10 & 16#ff00;
    //Dies ist eine mit WAITFORCONDITION auszuwertende
    //Bedingung (Rückgabewert der Funktion). Wenn FALSE,
    //wird die betreffende Task ausgesetzt, bis Bedingung
    //TRUE ergibt.
    myExpression :=myloc <> 16#0100;
END_EXPRESSION

```

PROGRAM myProgram

```

...
//Aufruf des Befehls mit Namen der Expression
WAITFORCONDITION myExpression WITH TRUE DO
    // hier mind. eine Anweisung, wird hochprior ausgeführt
END_WAITFORCONDITION;
END_PROGRAM
END_IMPLEMENTATION

```

FIG 9